Encapsulamento

Bom dia Professora!!
Hoje eu vou falar um pouquinho sobre encpasulamento.
Vamos então supor que eu faça um programinha para um caixa eletrônico onde eu mostre para uma pessoa o seu saldo.

Temos então a classe Principal com a função main
Escrever na tela a classe Pricipal

```
public class Principal{

    public static void main (String [] args){


     }

}
```

E temos também a classe Conta representando a conta bancária, e aqui para simplificar eu coloquei somente três atritos, como o nome, cpf e o saldo que queremos mostrar para a pessoa, quanto ela tem na conta.

Escrever na tela a classe Conta

```
public class Conta{

        //Atributos
        public String nome;
        public String cpf;
        public Double saldo;

        public Conta(){

        }

        public Conta (String nome, String cpf, Double saudo){

            this.nome = nome;
            this.cpf = cpf;
            this.saldo = saldo;

        }

}
```

Ok! Se você vier aqui na classe Principal e criar um objeto do tipo Conta, como por exemplo, Conta c recebendo new Conta, Conta c = new Conta.

```
public class Principal{

    public static void main (String [] args){
```

```
        Conta c = new Conta();

    }

}
```

Perceba que eu estou instanciando um novo objeto na memória do tipo conta. Então a minha váriável c é um Conta, então eu posso fazer isto, por exemplo, c.nome, recebe o nome da pessoa, como por exemplo “Maria”, c.nome = “Maria” e o c.cpf recebe vai receber um cpf qualquer, c.cpf = “23145898105”, eu vou inventar um aqui. E o c.saldo, eu posso colocar como se ele estivesse recebendo 900.0. então eu posso mandar imprimir na tela cada um desses valores. E por fim vamos testar o programa

```
public class Principal{

    public static void main (String [] args){
        Conta c = new Conta();

         c.nome = “Maria”;
         c.cpf = “23145898105”;
         c.saldo = 900.0

        System.out.println(c.nome);
        System.out.println(c.cpf);
        System.out.println(c.sado);

    }

}
```

Observe que imprimiu o meu console o nome Maria, o cpf 23145898105. Esta perfeito! Onde é que agora o encapsulamento. O encapusulamento é um recurso da POO onde você consegue controlar o acesso a alguns dados. Então aqui, por exemplo, se eu deixar o meu saldo livre para alterção assim como está.

Mostrar

saldo = 900.0

Eu posso colocar o sado que eu quiser.

Mostrar

Saldo = 1900.0

Com issso eu posso gerar um estado de incistência no meu sistema, eu não posso livremente mudar o sado. O que podemos fazer neste caso então. Posso voltar na classe conta.

Mostra a classe Conta

```
public class Conta{
```

```
        //Atributos
        public String nome;
        public String cpf;
        private Double saldo;

        public Conta(){

        }

        public Conta (String nome, String cpf, Double saudo){

            this.nome = nome;
            this.cpf = cpf;
            this.saldo = saldo;

        }

}
```

E trocar o nível de acesso do saldo para privado, private, que significa acesso privado. Se eu fizer isso e salvar. Perceba o que vai acontecer! Aqui no programa principal, vai acusar um erro.

Mostrar o programa principal

```
public class Principal{

    public static void main (String [] args){
            Conta c = new Conta();

             c.nome = “Maria”;
             c.cpf = “23145898105”;
             c.saldo = 900.0

            System.out.println(c.nome);
            System.out.println(c.cpf);
            System.out.println(c.sado);

    }

}
```

O compilador vai mostrar que este campo saldo está inacessível, isto porque este nível de acesso private permite que o dado seja alterado somente dentro da própria classe e não como a classe do programa principal

Selecionar todo o código para mostrar toda a classe.

Bom! Se agora o dado não é mais acessível como é que eu faço para obter ou para alterar este dado? É aí que entra os métodos get e set, onde get significa obter, e set signifia definir. Então por exemplo na classe Conta, além da definição dos atributos eu vou ter a definição também de um método para obter este saldo e por concenção nós iremos utilizar o método get.

```
public class Conta{

        //Atributos
        public String nome;
        public String cpf;
        private Double saldo;

        public Conta(){

        }

        public Conta (String nome, String cpf, Double saudo){

             this.nome = nome;
             this.cpf = cpf;
             this.saldo = saldo;

        }

       public getSaldo(){

      }

}
```

Como é que vai ficar? Este método será público, porque ele deverá ser acessado por outra classe. Depois do public eu vou dizer o tipo de dado que ele vai retornar, que no caso vai ser Double, porque eu quero retornar o saldo e eu vou colocar o método desta forma getSaldo, onde o S será maiúsculo. Esta é uma convenção da plataforma Java, onde você colocar o get em mínúculo e o S de saldo que é o atributo começando em maiúsculo. Vou colocar os parênteses indicando que não terá nenhum paramentro e no corpo do método vou colocar um comando para retornar o valor do sado que está aqui definido como atributo da classe. Agora eu vou salvar o programa e vou voltar para o programa principal.

```
public class Principal{

    public static void main (String [] args){
            Conta c = new Conta();

             c.nome = “Maria”;
             c.cpf = “23145898105”;
             //c.saldo = 900.0

             System.out.println(c.nome);
             System.out.println(c.cpf);
             System.out.println(c.sado);

    }

}
```

Por enquanto eu não posso alterar o valor. Então eu vou comentar este dado aqui c.saldo = 900.0. E

aqui no System.out.println(c.sado), na hora de imprimir eu vou colocar desta maneira System.out.println(c.getSaldo()). Perceba que agora eu consigo acessar o saldo por meio do método get, não mais diretamente o saldo. Vou testar novamente o programa. Perceba que ele mostra o valor null para o saldo. Porque este valor está nulo? É porque não atribuimos valor algum para o saldo, diferentemente dos valores atriuidos para o nome e cpf. Então o que eu posso fazer agora? Eu posso alterar o saldo ou posso fazer o seguinte. Vou definir na minha classe Conta.

```
public class Conta{

        //Atributos
        public String nome;
        public String cpf;
        private Double saldo;

        public Conta(){

           saldo = 0.0

        }

        public Conta (String nome, String cpf){

             this.nome = nome;
             this.cpf = cpf;
             saldo = 0.0;

        }

       public getSaldo(){

      }

}
```

Dentro do construtor dela, ou seja, o método que intancia o objeto. Vou falar que o meu saldo por padrão aqui no caixa vai começar com 0.0. Eu també posso inclusive fazer o mesmo no outro construtor. Onde antes eler recebia os três valores, só que agora ele não vai mais receber o saldo como argumento, eu não vou informar o saldo na hora de instanciar o objeto e na hora de intancia eu vou falar que o saldo recebe 0.0 também neste construtor. Se você quiser aproveitar a lógica do outro construtor, ao invés de repetir o comando aqui que é não é uma boa prática, você apaga esta repetição do código, onde não queremos repeti-lo e você cola simplesmente a palavra this() para referenciar o próprio objeto. Ou seja, eu estou chamando a lógica do construtor que não têm argumentos, que no caso é este daqui. Então ele primeiro vai executar o primeiro método, para depois executar o outro. Vamos então voltar ao programa principal.

```
public class Principal{

     public static void main (String [] args){
             Conta c = new Conta();

               c.nome = “Maria”;
               c.cpf = “23145898105”;
```

```
            //c.saldo = 900.0

            System.out.println(c.nome);
            System.out.println(c.cpf);
            System.out.println(c.sado);

    }

}
```

Criamos a váriável c aqui, atribuimo o nome e o cpf, vou apagar o comentário c.sado = 900.0. E vou executar a minha aplicação. Perceba que agora o saldo foi inicializado com 0.0. E agora para alterar o saldo como é que eu faço? Podemos fazer de duas formas! A primerira de forma básica! Eu venho aqui na classe Conta.

```
public class Conta{

      //Atributos
      public String nome;
      public String cpf;
      private Double saldo;

      public Conta(){

         saldo = 0.0

      }

      public Conta (String nome, String cpf){

          this.nome = nome;
          this.cpf = cpf;
          saldo = 0.0;

      }

     public getSaldo(){

        return saldo;

    }

    public void setSado(double slado){

            this.saldo = saldo;
     }

}
```

Então agora vou criar o método set, para eu alterar o valo do saldo. Vamos lá! Como é que é que eu faço para alterar o valor. Vamos ver a palavra-chave aqui do Java. Você vai colocar public, porque tanto o método get e set terá que ser público porque eles serão acessados por outras classes. No caso

do set é um método que não retorna nada ele só vai alterar o atributo do objeto, então eu vou colocar aqui void setSaldo e entre parênteses vou colocar o parâmentro que este argumento terá que receber, que é o dado que eu irei atribuir ao saldo. Então por exempo eu vou criar um paramentro chamado double saldo. Dentro do corpo do método eu vou colocar o seguinte this.saldo que é o atributo do objeto vai receber o sado sem a palavra this que é o saldo que eu coloquei no método. Então o que vai acontecer, se eu salvar isto aqui e ir ate o programa principal.

```
public class Principal{

    public static void main (String [] args){
            Conta c = new Conta();

             c.nome = “Maria”;
             c.cpf = “23145898105”;
             c.setSaldo = 100.0

            System.out.println(c.nome);
            System.out.println(c.cpf);
            System.out.println(c.sado);

    }

}
```

E colocar aqui a variável c.setSaldo recebedo por exemplo 100.0. Vou executar o programa. Repare que o console mostra agora que o saldo está valendo 100.0. Agora você pode me pergutra, eu apenas troquei os métodos get e set pela atribuição direta, eu estou fazendo a mesma coisa e compliquei sem necessidade. Bom! Não é bem assim! Porque dentro do método a classe Conta.

```
public class Conta{

        //Atributos
        public String nome;
        public String cpf;
        private Double saldo;

        public Conta(){

           saldo = 0.0

        }

        public Conta (String nome, String cpf){

            this.nome = nome;
            this.cpf = cpf;
            saldo = 0.0;

        }

       public getSaldo(){
```

```
    }

    public void setSado(double slado){


     }

}
```

No caso do getSaldo e do setSaldo, você pode colocar outra lógica adicional. Você pode ter uma regra no seu caixa eletronico que quando você muda o saldo, por explo, você muda uma taxa do cliente, aí você já coloca a lógica aqui também. Outro exemplo, eu não posso colocar o saldo como negativo então eu posso colocar uma lógica aqui, como por exemplo, if (sado < 0.0) aí vou fazer uma lógica aqui onde vou lançar uma exceção este tipo de coisa. Então o método te abre mais possibilidade para você encapsular alguma lógica dentro da operaçao de objter ou de definir o valor de um atributo.. Para finalizar vou colocar mais uma consideração. Eu posso ainda deixar a sematica mais apropriada ao inves de você criar um método padrão que é o setSaldo, que seria um método para atribuir diretamente um valor para o saldo, você pode criar operações que tem haver como o negócio, como por exemplo aqui que é uma conta bancária, eu posso depositar e posso sacar valores nesta conta.

```
public class Conta{

      //Atributos
      public String nome;
      public String cpf;
      private Double saldo;

      public Conta(){

         saldo = 0.0

      }

      public Conta (String nome, String cpf){

           this.nome = nome;
           this.cpf = cpf;
           saldo = 0.0;

      }

     public getSaldo(){

        return saldo;

    }

    public void depositar(double valor){

                saldo = saldo + valor;
     }
```

```
        public void sacar(double valor){

                saldo = saldo - valor;
        }
}
```

Neste caso eu vou criar o método depositar que vai recer aqui um valor. O que seria depositar um valor na minha conta? Seria fazer o saldo, onde neste caso eu poderia tirar o this, porque que não há mais ambiguidade com o nome do atributo, eu posso fazer com que o saldo possa receber o que já tinha nele + o valor. Então eu só posso mudar o saldo por meio da operação de depositar, onde representa uma lógica do meu negócio. Da mesma forma você pode criar a operação de sacar, sacar o dinheiro. Então eu vou na verdade é subtrair este valor da conta. E ai mais uma vez você poderá adicionar uma lógica. Você pode testar para ver se têm saldo. Então eu posso fazer assi, if (saldo < valo), se o saldo for menor que o valor que eu quero sacar eu posso lançar uma exceção para não dizer sacar. Então este são alguns principios do encapsulamento,que você pode implementar na sua classe para que ela fique com um objeto consistente e protegido de operaçãoes que poderia gerar um estado inválido para o seu sistema. Repare que do jeito que está aqui eu não posso ir no programa Principal e fazer o getSaldo.

```
public class Principal{

    public static void main (String [] args){
            Conta c = new Conta();

             c.nome = “Maria”;
             c.cpf = “23145898105”;
             c.depositar = 100.0
             c.sacar = 20.0

            System.out.println(c.nome);
            System.out.println(c.cpf);
            System.out.println(c.sado);

    }

}
```

Então eu vou depositar 100.0 e vou sacar 20.0. Vou salvar o programama e vou executar. Se por padrão o saldo começa com 0.0, eu depositei 100.0 e saquei 20.0.Quanto vai ser o saldo no final? Vai ter que mostrar 80.0 no cosole. Vamos então executar o programa! Repare que o console motrou realmente 80.0. Então é isso! Obrigado!